

本节内容

# 浮点数

加减运算  
强制类型转换

关注公众号【研途小时】获取后续课程完整更新！

王道考研/CSKAOYAN.COM

# 本节总览



浮点数的运算

加减运算

强制类型转换

# 浮点数的加减运算

浮点数加减运算步骤:

$$9.85211 \times 10^{12} + 9.96007 \times 10^{10}$$

① 对阶

思考为什么  
是小阶向大  
阶靠齐?

$$\textcircled{1} 9.85211 \times 10^{12} + 0.0996007 \times 10^{12}$$

计算机内部, 尾  
数是定点小数

② 尾数加减

$$\textcircled{2} 9.9517107 \times 10^{12}$$

③ 规格化

③ 如果尾数加减出现类似  $0.0099517 \times 10^{12}$  时, 需要“左规”; 如果尾数加减出现类似  $99.517107 \times 10^{12}$  时, 需要“右规”

④ 舍入

可以有不同  
的舍入规则

④ 若规定只能保留6位有效尾数, 则

$$9.9517107 \times 10^{12} \rightarrow 9.95171 \times 10^{12}$$

(多余的直接砍掉)

或者,  $9.9517107 \times 10^{12} \rightarrow 9.95172 \times 10^{12}$  (若砍掉部分非0, 则入1)

或者, 也可以采用四舍五入的原则, 当舍弃位 $\geq 5$ 时, 高位入1

⑤ 判溢出

⑤ 若规定阶码不能超过两位, 则运算后阶码超出范围, 则溢出

$$\text{如: } 9.85211 \times 10^{99} + 9.96007 \times 10^{99} = 19.81218 \times 10^{99}$$

规格化并用四舍五入的原则保留6位尾数, 得  $1.98122 \times 10^{100}$

阶码超过两位, 发生溢出 (注: 尾数溢出未必导致整体溢出, 也许可以通过③④两步来拯救)

关注公众号【研途小时】获取后续课程完整更新!

# 浮点数的加减运算

例：已知十进制数 $X=-5/256$ 、 $Y=+59/1024$ ，按机器补码浮点运算规则计算 $X-Y$ ，结果用二进制表示，浮点数格式如下：阶符取2位，阶码取3位，数符取2位，尾数取9位

用补码表示阶码和尾数

扩展：11.011000000

双符号位补码：11.011 双符号位补码：11011

补码：1.011

补码：1011

0. 转换格式

$5D = 101B$ ,  $1/256 = 2^{-8} \rightarrow X = -101 \times 2^{-8} = -0.101 \times 2^{-5} = -0.101 \times 2^{-101}$

$59D = 111011B$ ,  $1/1024 = 2^{-10} \rightarrow Y = +111011 \times 2^{-10} = +0.111011 \times 2^{-4} = +0.111011 \times 2^{-100}$

X: 11011, 11.011000000 Y: 11100, 00.111011000

浮点数加减运算步骤：

1. 对阶
2. 尾数加减
3. 规格化
4. 舍入
5. 判溢出

# 浮点数的加减运算

例：已知十进制数 $X=-5/256$ 、 $Y=+59/1024$ ，按机器补码浮点运算规则计算 $X-Y$ ，结果用二进制表示，浮点数格式如下：阶符取2位，阶码取3位，数符取2位，尾数取9位

用补码表示阶码和尾数

0. 转换格式

$$5D = 101B, 1/256 = 2^{-8} \rightarrow X = -101 \times 2^{-8} = -0.101 \times 2^{-5} = -0.101 \times 2^{-101}$$

$$59D = 111011B, 1/1024 = 2^{-10} \rightarrow Y = +111011 \times 2^{-10} = +0.111011 \times 2^{-4} = +0.111011 \times 2^{-100}$$

X: 11011,11.011000000    Y: 11100,00.111011000

浮点数加减运算步骤：

1. 对阶 使两个数的阶码相等，小阶向大阶看齐，尾数每右移一位，阶码加1

① 求阶差： $[\Delta E]_{\text{补}} = 11011 + 00100 = 11111$ ，知 $\Delta E = -1$

② 对阶：X: 11011,11.011000000  $\rightarrow$  11100,11.101100000     $X = -0.0101 \times 2^{-100}$

2. 尾数加减

3. 规格化

4. 舍入

5. 判溢出

# 浮点数的加减运算

例：已知十进制数 $X=-5/256$ 、 $Y=+59/1024$ ，按机器补码浮点运算规则计算 $X-Y$ ，结果用二进制表示，浮点数格式如下：阶符取2位，阶码取3位，数符取2位，尾数取9位

用补码表示阶码和尾数

0. 转换格式

$$5D = 101B, 1/256 = 2^{-8} \rightarrow X = -101 \times 2^{-8} = -0.101 \times 2^{-5} = -0.101 \times 2^{-101}$$

$$59D = 111011B, 1/1024 = 2^{-10} \rightarrow Y = +111011 \times 2^{-10} = +0.111011 \times 2^{-4} = +0.111011 \times 2^{-100}$$

X: 11011,11.011000000    Y: 11100,00.111011000

浮点数加减运算步骤：

1. 对阶 使两个数的阶码相等，小阶向大阶看齐，尾数每右移一位，阶码加1

① 求阶差： $[\Delta E]_{补} = 11011 + 00100 = 11111$ ，知 $\Delta E = -1$

② 对阶：X: 11011,11.011000000  $\rightarrow$  11100,11.101100000     $X = -0.0101 \times 2^{-100}$

2. 尾数加减 -Y: 11100,11.000101000

X-Y: 11100, 10.110001000

11.101100000

+ 11.000101000

10.110001000

X-Y

$$= (-0.0101 \times 2^{-100}) - (+0.111011 \times 2^{-100})$$

$$= (-0.0101 - 0.111011) \times 2^{-100}$$

$$= -1.001111 \times 2^{-100}$$

3. 规格化

4. 舍入

5. 判溢出

# 浮点数的加减运算

例：已知十进制数 $X=-5/256$ 、 $Y=+59/1024$ ，按机器补码浮点运算规则计算 $X-Y$ ，结果用二进制表示，浮点数格式如下：阶符取2位，阶码取3位，数符取2位，尾数取9位

用补码表示阶码和尾数

0. 转换格式

$$5D = 101B, 1/256 = 2^{-8} \rightarrow X = -101 \times 2^{-8} = -0.101 \times 2^{-5} = -0.101 \times 2^{-101}$$

$$59D = 111011B, 1/1024 = 2^{-10} \rightarrow Y = +111011 \times 2^{-10} = +0.111011 \times 2^{-4} = +0.111011 \times 2^{-100}$$

X: 11011,11.011000000    Y: 11100,00.111011000

浮点数加减运算步骤：

1. 对阶 使两个数的阶码相等，小阶向大阶看齐，尾数每右移一位，阶码加1

① 求阶差： $[\Delta E]_{补} = 11011 + 00100 = 11111$ ，知 $\Delta E = -1$

② 对阶：X: 11011,11.011000000  $\rightarrow$  11100,11.101100000     $X = -0.0101 \times 2^{-100}$

2. 尾数加减    -Y: 11100,11.000101000    11.101100000

X-Y: 11100,10.110001000    + 11.000101000

10.110001000

3. 规格化

X-Y: 11100,10.110001000  $\rightarrow$  11101,11.011000100

4. 舍入    无舍入

5. 判溢出    常阶码，无溢出，结果真值为 $2^{-3} \times (-0.1001111)_2$

X-Y

$$= (-0.0101 \times 2^{-100}) - (+0.111011 \times 2^{-100})$$

$$= (-0.0101 - 0.111011) \times 2^{-100}$$

$$= -1.001111 \times 2^{-100}$$

$$= -0.1001111 \times 2^{-011}$$

# 浮点数的加减运算-舍入

有的计算机可能会把浮点数的尾数部分单独拆出去计算(24bit→32bit)，算完了经过舍入(32bit→24bit)再拼回浮点数

**“0”舍“1”入法：**类似于十进制数运算中的“四舍五入”法，即在尾数右移时，被移去的最高数值位为0，则舍去；被移去的最高数值位为1，则在尾数的末位加1。这样做可能会使尾数又溢出，此时需再做一次右规。

**恒置“1”法：**尾数右移时，不论丢掉的最高数值位是“1”还是“0”，都使右移后的尾数末位恒置“1”。这种方法同样有使尾数变大和变小的两种可能。

浮点数加减运算步骤：

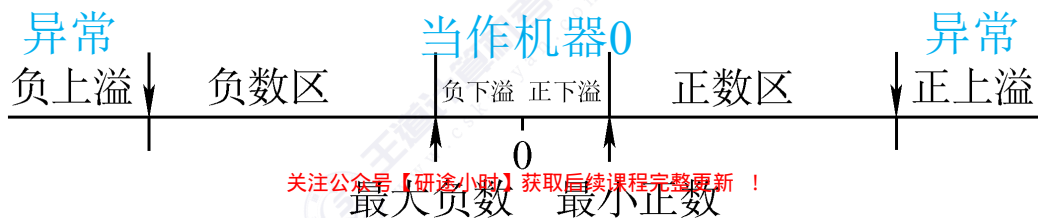
1. 对阶
2. 尾数加减
3. 规格化
4. 舍入
5. 判溢出

如：加减结果为11100,10.110001011

0舍1入：11100,10.110001011 → 11101,11.011000101 1  
→ 11101,11.011000110 1

恒置1:11100,10.110001011 → 11101,11.011000101 1  
→ 11101,11.011000101 1

右规时就会面临舍入的问题



关注公众号【研途小时】获取后续课程完整更新！



# 强制类型转换

类型	16位机器	32位机器	64位机器
char	8	8	8
short	16	16	16
int	16	32	32
long	32	32	64
long long	64	64	64
float	16	32	32
double	64	64	64

char → int → long → double

float → double

范围、精度从小到大，转换过程没有损失

32位

int: 表示整数，范围  $-2^{31} \sim 2^{31}-1$ ，有效数字32位

float: 表示整数及小数，范围  $\pm[2^{-126} \sim 2^{127} \times (2-2^{-23})]$ ，有效数字23+1=24位

int → float: 可能损失精度

float → int: 可能溢出及损失精度

# 本节回顾

## 浮点数的运算

### 加减运算

真值到机器数的转换

注意阶码、尾数采用什么码表示；注意符号扩展凑足规定位数

对阶

小阶向大阶看齐，尾数算数右移一位，阶码加1，直到阶码相同

注：对阶可能导致丢失末位精度

尾数加减

通常采用双符号位表示尾数，这样可以挽救尾数溢出

规格化

左规

尾数最高数值位为无效位时，尾数左移，阶码减1

右规

尾数双符号位不同时，尾数右移，阶码加1

舍入

尾数的位数有限导致的问题，常用方法：0舍1入、恒置1

溢出判断

阶码上溢

抛出异常（中断）

阶码下溢

按机器0处理

采用双符号位，  
可拯救尾数溢出

### C强制类型转换

无损

char—>int—>long—>double

float—>double

有损

int—>float

可能会损失精度（float尾数的数值位有1+23位）

float—>int

可能会溢出，也可能损失精度（如小数转整数）

关注公众号【研途小时】获取后续课程完整更新！

王道考研/CSKAOYAN.COM



公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研